Kódování a komprese dat 2023/2024: Projekt

Téma: Komprese obrazových dat s využitím Huffmanova kódování
Datum odevzdání: nejpozději do 12. 5. 2024 (včetně) prostřednictvím IS VUT.
Forma odevzdání: elektronicky - soubory v archívu ZIP
Počet bodů: max. 30
Poznámka: k zadání projektu je v IS VUT zveřejněna sada testovacích obrázků
Dotazy: vasicek@fit.vutbr.cz

Zadání:

Cílem projektu je v programovacím jazyce C/C++ vytvořit aplikaci pro kompresi šedo tónových obrazových dat, kde se uplatní princip Huffmanova kódování. Vytvořené řešení musí být spustitelné jako tzv. konzolová aplikace (tj. z příkazového řádku) pod OS Linux v prostředí počítačové sítě FIT VUT v Brně.

Podrobné pokyny k vypracování:

a) formát vstupních dat:

Vstupní data pro tuto aplikaci budou reprezentována sadou obrázků ve formátu RAW (surová data bez hlavičky a bez použití komprese). Jednotlivé obrazové body (pixely) těchto referenčních obrázků mohou nabývat hodnot odstínů šedi v intervalu 0 až 255. Z důvodu použití RAW formátu je tedy nutné aplikaci prostřednictvím parametru v příkazové řádce definovat velikost vstupního obrazu. Velikost vstupního obrazu se specifikuje jen v případě komprese. Pokud ji potřebujete pro dekompresi, je nutné uložit jako součást výstupních dat.

b) zpracování vstupních dat:

Serializace dat: Implementujte statickou a adaptivní metodu skenování vstupního obrazu (viz [2], skenování obrazových dat v části věnující se RLE), přičemž v případě adaptivní verze uvažujte minimálně dva typy průchodu: horizontální a vertikální. Typ průchodu volte v případě adaptivní verze tak, aby bylo dosaženo co nejvyšší kompresní účinnosti. Obraz pro zpracování rozložte na menší bloky o fixní velikosti, např. 16x16 pixelů. V případě statické verze procházejte vstupní obraz horizontálně a celý vstup chápejte jako jeden souvislý blok dat, tzn. neprovádějte dělení na menší bloky.

Transformace dat: Na základě parametru v příkazovém řádku pracujte se serializovanými vstupními obrazovými daty dvěma různými způsoby. Buď berte v potaz *přímo hodnotu samotných pixelů* nebo použijte vhodný *model umožňující zvýšit kompresní účinnost*. Jako model lze použít např. diference sousedních pixelů (viz úvodní slidy KKO), případně jakýkoliv jiný pokročilejší kontextově orientovaný přístup. Za tímto krokem bude následovat technika RLE, jejímž cílem bude zvýšit kompresní účinnost. Volba varianty RLE závisí na Vás. V případě, že není možné dosáhnout pro daný blok komprese, uložte data nekomprimovaně.

Každý blok si tedy bude muset nést ve zkomprimovaných datech pomocnou informaci o způsobu zpracování bloku (zvolená varianta skenování, použití komprese)

c) funkce a uživatelské rozhraní:

Výsledné řešení bude mít charakter aplikace spustitelné z příkazového řádku pomocí příkazu *huff_codec*, který získáme po překladu zdrojových kódů vlastní implementace. Funkci aplikace bude možné řídit pomocí sady přepínačů či parametrů zadávaných na příkazovém řádku:

-c volba režimu, aplikace bude vstupní soubor komprimovat,
-d volba režimu, aplikace bude vstupní soubor dekoprimovat,
-m aktivuje model a RLE (viz bod b) zadání) pro předzpracování vstupních dat
-a aktivuje režim adaptivního skenování obrazu (viz bod b) zadání); jinak je použito sekvenční skenování v horizontálním směru bez dělení na bloky
-i <*ifile***>** název vstupního souboru. Dle režimu činnosti aplikace, viz přepínače –c/-d výše, se

jedná buď o data určená ke kompresi nebo dekompresi.
-o <*ofile***>** název výstupního souboru. Dle režimu činnosti aplikace, viz přepínače –c/-d výše, se jedná buď o zkomprimovaná či dekomprimovaná data,
-w <*width_value*> udává šířku obrazu, přičemž platí vztah *width_value* >= 1. Parametr se zadává pouze v případě komprese, aby aplikace věděla, jak široký obraz je. Pro potřeby dekomprese je nutné uložit všechny potřebné informace do výstupního souboru.
-h vypíše nápovědu na standardní výstup a ukončí se.

Huffmanovo kódování lze implementovat v libovolné verzi, avšak doporučena je kanonická verze.

d) komentáře zdrojových kódů:
Zdrojový kód vhodně komentujte. Komentujte zejména parametry a činnost funkcí, datové typy, lokální proměnné (pokud to nebude z názvu přímo vyplývat).

Každý zdrojový soubor opatřete hlavičkou, ve které bude datum vytvoření, název a stručný popis souboru a informace o autorovi – jméno, příjmení a login.

e) kompilátor a prostředí:
Odevzdané řešení musí být možné přeložit/spustit na serveru ***merlin.fit.vutbr.cz***, kde bude také probíhat ověření funkčnosti a měření výkonnosti aplikace (viz odstavec g) zadání).

f) dokumentace:
Součástí odevzdaného řešení bude stručná dokumentace, která bude na začátku obsahovat stručný rozbor řešeného problému (není žádoucí opisovat dlouhé teoretické statě z odborné literatury či studijních materiálů). Dále nástin vlastního řešení včetně např. vysvětlení funkce důležitých datových struktur a dalších relevantních aspektů. Nezapomeňte v dokumentaci uvést i odkazy na použité informační zdroje, z nichž jste čerpali během řešení projektu, nejedná-li se o přednášky.

Povinnou součástí dokumentace je taktéž vyhodnocení efektivity v těchto režimech činnosti:
- Komprese využívající statické skenování obrazu, bez modelu
- Komprese využívající statické skenování obrazu, s modelem (tj. parametr -m)
- Komprese využívající adaptivní skenování obrazu, bez modelu (tj. parametr -a)
- Komprese využívající adaptivní skenování obrazu, s modelem (tj. parametry -a -m)

Efektivitu komprese vyjádřete coby průměrný počet bitů potřebný na zakódování jednoho pixelu obrazu a času komprese ve výše uvedených režimech činnosti nezávisle pro jednotlivé případy obrazových dat. Naměřená data prezentujte formou tabulky. Pro každý obraz spočtěte entropii zdroje, budeme-li uvažovat, že výskyty hodnot jsou izolované a navzájem nezávislé. Uveďte také průměrný počet bitů pro jednotlivé metody vypočtený přes všechna přiložená data.

Rozsah dokumentace by neměl přesáhnout 4 strany formátu A4.

g) odevzdává se (checklist):
Archív ve formátu ZIP, který bude mít následující obsah:
- zdrojové soubory vlastní implementace v jazyce C/C++
- Makefile soubor (zavoláním *make* musí vzniknout spustitelný soubor *huff_codec*)
- dokumentaci ve formátu PDF

Použití externích knihoven není dovoleno s výjimkou libdivsufsort. Použijete-li knihovnu, nezapomeňte ji umístit do odevzdávaného archívu. Makefile konstruujte tak, aby volal pouze příkazy související s GCC/G++, nikoliv cmake, příkazy shellu apod.

h) bonusové body:
Prvních 5 řešitelů s nejefektivnější implementací z pohledu dosažené míry komprese a rychlosti komprese a dekomprese má možnost získat až 3 bonusové body.

Literatura:
[1] Salomon, D.: "Data Compression, The Complete Reference," NY, USA, Springer, 2000
[2] Přednášky k předmětu Kódování a komprese dat

Tipy a rady na závěr:
- Důležitou vlastností je přenositelnost. Pro různé platformy může být velikost typů int, long různá. Těmto problémům se vyhneme použitím předdefinovaných typů *int8_t, int16_t, int32_t, u_int_8_t* apod., které jsou definovány v knihovně sys/types.h.
- Pro zpracování parametru příkazové řádky je výhodné využít funkce *getopt* (resp. *getopt_long*), jejíž rozhraní je definováno v unistd.h (resp. getopt.h).
- Pro implementaci některých datových struktur a operací s nimi může být výhodné použít v aplikaci knihovnu STL (C++ Standard Template Library).
- Pro ladění potíží s pamětí (Segmentation fault apod.) doporučuji použít nástroj *valgrind*, který se používá na aplikaci zkompilovanou pomocí *make debug* (příp. *gcc -ggdb3*).